SVL8-2003-0140 Object Activation and Monitoring Service using MQ as a triggering mechanism

(shutdown)<near/16>(restart OR reboot)<near/16>(program)

√JP2002111674A2: METHOD AND SYSTEM FOR MANAGING CONNECTION AND COMPUTER READABLE RECORDING MEDIUM WITH PROGRAM FOR REALIZING THAT METHOD RECORDED THEREON

PROBLEM TO BE SOLVED: To make possible to identify a normally communicable connection between server processes when an extra server machine restarts after shutdown in a client server system performing online transaction where the function of the server is distributed to a plurality of server machines.
SOLUTION: When a server process establishes connection with the server process of an extra server machine, the time acquired from a system is registered as a connection establishment time in a table 11 managing connection. A normally communicable connection can thereby be identified by checking the connection establishment time even when the extra server machine restarts after shutdown.

√US20020073409A1: Telecommunications platform with processor cluster and method of operation thereof
√WO0248886A2: TELECOMMUNICATIONS PLATFORM WITH PROCESSOR CLUSTER AND METHOD OF OPERATION THEREOF

In its various aspects, the cluster support function (50) includes a state storage system (200) and a name server system (300) for facilitating restart and switch over of programs and programs executed by processors of the cluster.

. . .

In one aspect, the cluster support function includes a state storage system. An active version of a program executing on a first processor of the cluster stores state data in the state storage system. The state data is sufficient for a standby version of the program to resume operation (e.g., on another processor) should the active version of the program terminate, or for the same version of the program to restart. In the event one of upgrade or shutdown of the active version of the program, the another version of the program can be the standby version thereof which executes on a second processor of the cluster. The state data of the event-affected program is provided to the another (standby) version of the program for resumption of operation of the program.

. . .

The stored data can be utilized in two different situations. A first situation occurs when a processor (and/or program) is restarted, likely due to a software error. A second situation occurs when a hardware fault is detected and the processor is taken out of operation. In the first situation, the program that stored the data in the state storage system 200 is restarted and will then fetch the data from the state storage system 200 and thereafter continue execution at the point indicated by the newly fetched state. In the second situation, a standby version of the program (e.g., the standby program) will be activated (see, e.g., start message 6-2' in Fig. 6B) on another processor, with the standby program fetching the data from state storage system 200 and resuming operation of the program on the other processor.

✓US6591379: Method and system for injecting an exception to recover unsaved data

Program module exception injecting method, involves executing data recovery program to notify failure reporting executable program and restarting program module to display recovered unsaved data in user interface

. . .

A method and system for injecting an exception into a hung program module to recover unsaved data. The method and system allows a user to determine that the program module is hung and inject an exception into the hung program module. In response to the injected exception, program code, within the program module, is activated to detect the crash and launch a failure reporting executable program that instructs the hung program module to execute its data recovery program. The file recovery program recovers the unsaved data and notifies the failure reporting executable program when it has completed the data recovery. Once the failure reporting executable program receives the notification, it will instruct the operating system to restart the program module. The restarted program module then displays a user interface with a list of the unsaved data that were recovered.

✓JP2003131896A2: INFORMATION PROCESSING DEVICE, RESTARTING METHOD AND RESTARTING PROGRAM

PROBLEM TO BE SOLVED: To provide an information processing device by which a program can be normally restarted under the same operating environment.
SOLUTION: In the device, a flash memory 17 is provided as a built-in disk apart from RAM 18 for program work, and setting information and patch file stipulating the operating environment of the program are stored to this flash memory 17. When restarting is instructed by pushing a reset key 14, the RAM 18 is initialized and setting information and the patch file read from the flash memory 17 are written in. Thus, the program can be restarted under the same operating environment as before.

WO0193038A3: SCALABLE JAVA SERVERS FOR NETWORK SERVER APPLICATIONS

A monitoring program restarts the Distributor if it fails. The system thus provides a scalable and available set of Java-implemented Application Servers without requiring any change to the Java-implemented application itself.

. . .

Lack of availability means that the Java environment does not provide a way to continue the processing of the Java Application if the Application or the JVM encounters a fatal error. In this case, manual intervention is required to reestablish the Java environment and restart the failed application, resulting in a severe interruption to the service provided by the Java program. Given the critical nature of the Java e-commerce applications, the lack of scalability and availability in the Java environment is simply unacceptable. Thus, there is a need for a modified Java environment which supplies scalability and availability without sacrificing portability and architectural-neutral features of Java technology.

. . .

Yet another advantage is that Java-implemented Application Servers can be used in environments that demand little or no down time. An Application Server that fails is restarted on its original CPU or another CPU to maintain a given number of Application Servers present to service client requests.

. . .

Distributor Restart
FIG. 9 shows a flow chart for the restarting of the distributor module. If the Distributor terminates because of a processor element failure, the PATHMON module restarts the Distributor, which then performs the steps in the process of reading the configuration parameters. This process includes beginning a dialog with each static Application server and sending an InitialMsg message to each Application server. If the server has already received the message, the server assumes that the Distributor is restarting and reply with a RestartReply message. The Distributor updates the ServerStatus structure for the Application Server and continues processing in the operation phase.

JP2001022709A2: CLUSTER SYSTEM AND COMPUTER-READABLE STORAGE MEDIUM STORING PROGRAM

PROBLEM TO BE SOLVED: To easily introduce a new program into a cluster system and to continuously execute the program even if some abnormality occurs.

SOLUTION: In the cluster system which monitors the operation states of programs 6a to 6c running on computers 2a and 2b, an identification information acquiring means 11a acquires identification information of the programs 6a to 6c and a monitor means 12a monitor whether or not the programs 6a to 6c indicated by the acquired identification information are normal. When it is judged that a monitored program is abnormal, a restarting means 12b restarts the abnormal program on the computer 2a where the abnormal program was executed. Once the abnormality of the restarted program is detected, a program transfer means 11b executes the programs 6a to 6c on the computer 2a where the abnormal program was executed, on the other computer 2b.

US6031991: Debug system and method for reproducing an error occurring in parallel-executed programs

While processing the programs 2a, 2b and 2c, the processors 1a, 1b and 1c take check points in regular intervals of time. Data taken at each check point are sequentially stored in a storing section 3.

Each processor generates a trap signal when an error has occurred in a corresponding program. A detecting section 4 detects trap signals generated from any of the processors 1a, 1b and 1c. Upon detection of a trap signal, the detecting section 4 instructs a restart-executing section 5 to perform restart processing.

The restart-executing section 5 performs the restart processing in accordance with the instruction from the detecting section 4.

JP5165539A2: AUTOMATIC RESTART CONTROL SYSTEM FOR BATCH PROGRAM

PURPOSE: To continue the processing of a batch program without any manual work by acquiring the condition of executing the batch program as history information for saving labor in the case of managing the execution of the batch program, recovering an interrupted processing up to a synchronized point when the execution of the program is interrupted, and instructing a restarting point from the history information to the restarted program so as to restart the program after judging whether the restart is enabled or not.

CONSTITUTION: The communication of the batch program from a program A11 through an adapter 12 to a control program 13 is processed for automatically restarting the batch program. The control program 13 records the executed result of the program A11 in a processing object data set 14 and a history data set 15 and based on this information, the program is automatically restarted.

JP2224053A2: MONITORING SYSTEM FOR INPUT TIME OF TERMINAL

PURPOSE: To continue the processing without ending conversation even in the case of a response time becomes longer by setting in advance an operation at the time of a time sharing system TSS command execution program for expediting an input having possibility that the response time becomes longer at the time of exceeding an input time monitoring timer value, and also, restoring it thereafter.

CONSTITUTION: An operation change declaring means 12 declares a change of an operation of a TSS command execution program at the time of a response time exceeds an input time monitoring timer value of a terminal, an operation change informing means 13 informs a change of the operation of the TSS command execution program, and an input time monitoring state informing means 14 informs to the TSS command execution program a fact that it exceeds the input time monitoring timer value. A TSS command execution program restarting means 15 restarts the TSS command execution program, and an operation restoration declaring means 16 declares a restoration of the operation of the TSS command execution program at the time of exceeding the input time monitoring timer value. In such a way, even in the case when the response time becomes long, the processing can be continued.

JP58195345A2: LINE ADAPTOR

PURPOSE: To reduce the load required for the formation of transmission control program and the hardware, by detecting the stop of program to a line of itself and restarting the program, therefore accommodating lines in mixture for the different transmission control procedure and communication speed.

CONSTITUTION: A line adaptor LA' contains a buffer register, a shift register and a restart request circuit 111. When a timeout signal 206 is generated on the way of the transmission control program, a timeout detection circuit 207 detects whether or not it is the timeout to the line of itself, a transmission timeout signal 205 is transmitted, an FF 204 is set, and a program restart request signal 203 at the transmission side being the output of the FF 204 outputs a program execution request signal 201 at the transmission side to a communication control section CPC again. Further, the CPU stops the program of the said line once at the same time when the signal 206 is generated, and the processing is transferred to the program processing of the line when the request of execution from the line having higher priority than the said line exists, and the program of the said line is restarted with a request signal 201 after the end of the processing.